What can an extended SDAI do for a new Integration Model ?

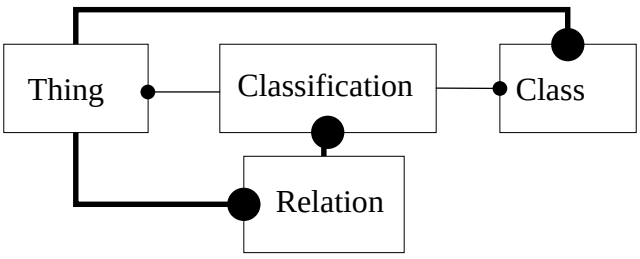
Assumption: Express, part21, SDAI is going to be used for this.

- Usage of the CONNOTATIONAL SUBTYPE of Express2
- Overview of SDAI with a the perspective of the complete Java binding (WG11/N060)
- The mapping extensions of the SDAI_dictionary_schema and associated operators
  - WG11/N050 SDAI Mapping Schema
  - WG11/N051 SDAI Mapping Schemas II
  - WG11/N068 Requirements for the next version of SDAI
  - WG11/N075 Mapping Operations for the SDAI
- Data conversion: AIM <=> ARM <=> EPISTLE
  but not: AIM <=> EPISTLE

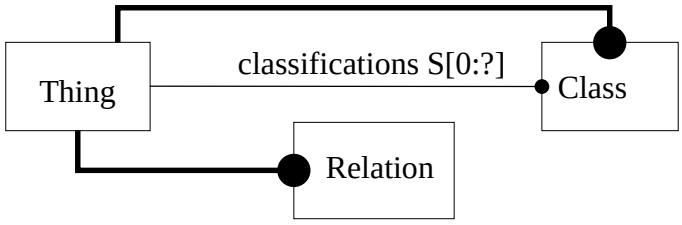
Lothar Klein, LKSoftWare GmbH, lothar.klein@lksoft.com
1999-04-22 ISO TC184/SC4/WG10

Simplify the implementation of classification

From:



To:



Assumption: Classification of a thing needs not to be not referenced

Application Model and Data

ENTITY **engine**
    ...
END_ENTITY

ENTITY **car**
    ...
    motor: engine;
END_ENTITY

#5=engine(...)
#6=car(…, #5)

Use CONNOTATIONAL SUBTYPE

ENTITY **engine** CONNOTATIONAL SUBTYPE of (thing...)
DERIVE
    ...
END_ENTITY

ENTITY **car** CONNOTATIONAL SUBTYPE of (thing...)
DERIVE
    ...
    motor: engine = … ;
END_ENTITY

-- *A population of just this is meaningless:*
#5=thing(())
#6=thing(())

Relative classification by defining a pattern for which to search

ENTITY **engine** CONNOTATIONAL SUBTYPE of (thing...)
WHERE
wr1: // SELF.classifications[i]-> class, class.name = **'engine'**
END_ENTITY

ENTITY **car** CONNOTATIONAL SUBTYPE of (thing...)
WHERE
wr1: // SELF.classifications[i]-> class, class.name = **'car'**
DERIVE
motor: engine // SELF<-composition.whole
(composition->classification[i]->class, class.name=**'motor_of_car'**)
composition.part->engine
END_ENTITY

-- This is similar to mapping tables in APs where a pattern of data is defined.

#1=class((...), 'engine')
#2=class((...), 'truck')
#3=class((...), 'car')
#4=class((...), 'motor_of_car')

#5=thing((#1))
#6=thing((#2, #3))
#7=composition((#4), #5, #6)

-- Problem: What happens if there are several car classes?
#8=class((...), 'car')

Absolute classification

```
SCHEMA automobile;
REFERENCE FROM core_schema;
CONST engine_class : class := class();  -- #1
        car_class : class := class();       -- #2
        truck_class : class := class();   -- #3
        motor_of_car_class := class();  -- #4

ENTITY engine CONNOTATIONAL SUBTYPE of (thing...)
WHERE   wr1: // SELF.classifications[i] = engine_class
END_ENTITY

ENTITY car CONNOTATIONAL SUBTYPE of (thing...)
WHERE    wr1: // SELF.classifications[i] = car_class
DERIVE
        motor: engine // SELF<-composition.whole
          (composition->classification[i] = motor_of_car_class )
          composition.part->engine
END_ENTITY
```

Repository: *automobile_schema_data*

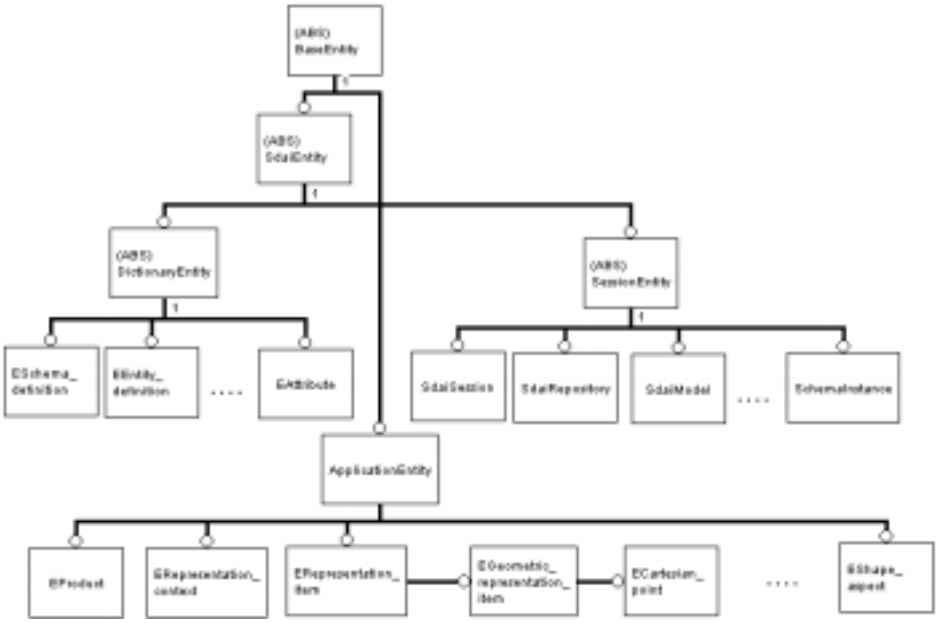
#1=class((...)) -- engine
#2=class((...)) -- truck
#3=class((...)) -- car
#4=class((...)) -- motor_of_car

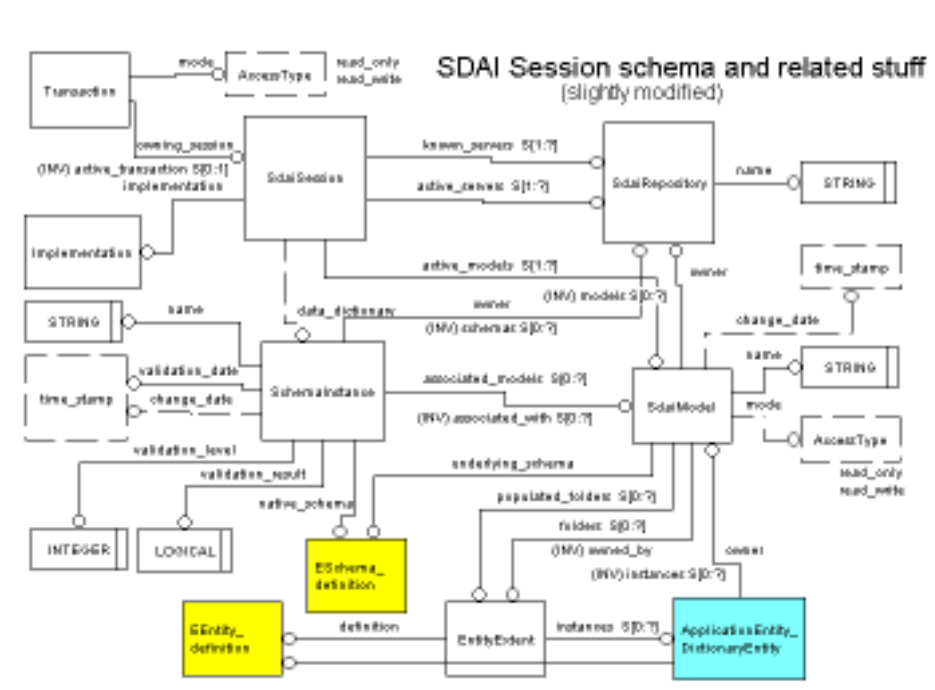
Repository: *user_data*

#5=thing((#1))
#6=thing((#2, #3))
#7=composition((#4), #5, #6)

Problem: Part21 needs to handle references to external repositories / files

Entity Hierarchy





Standard SDAI operators

Application instances:
- create, delete, validate

Attributes of Entity instances:
- test, get

Attributes of Application instances:
- set/create, unset

Members of aggregates:
- test, get, set/add/create, unset/remove

Special operators to work on Session instances:
- ...
- SchemaInstance - validate

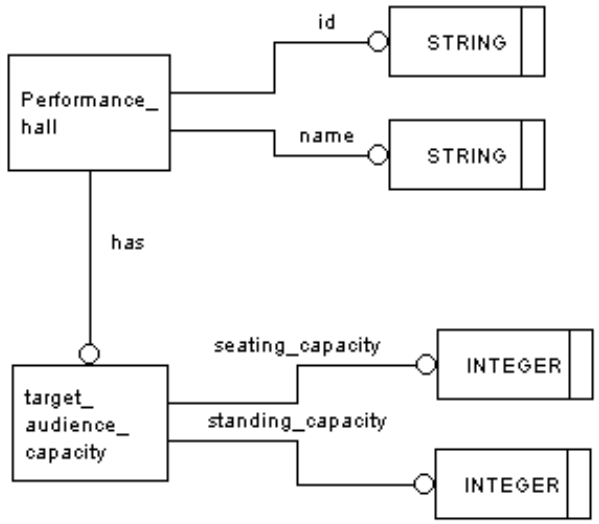
Proposed Extended SDAI operators

SdaiRepositories:
- import/export p21 files
- create, find/access on a network

Full featured SDAI_dictionary_data:
- expressions, functions, constants (PLIB 20)
- supertype constraints (ANDOR, AND, ONE OF)
- mapping information
- use dictionary entities like application entities

New Mapping operations

AP1 ARM EXPRESS (partially)
Mapping and AIM see: William F. Danner, WG10N...



Conclusion

Mapping of on AIM into an EPISTLE model is cumbersome because of the many intermediate entities which are introduced through the IRs (exception e.g. part42)

Therefor it is more suitable to map the ARM Express into an EPISTLE model.

Even for the “old” APs 201 and 203 prototype ARM-Express schemas already exist (WG10N107, …, Phil Kennicott)

Use merged Express-2/Express-X functionality do document the mapping and use this with new SDAI mapping functionality.